**Assignment 2**

Nicholas Andrusiw, 250966821

11/19/2021

**Equivalence Class Testing: getAgeFactor(clientAccount)**

|  |  |  |
| --- | --- | --- |
| *Requirement* | *clientAccount.age* | *ageFactor* |
| R1 | Age<15 | 0 |
| R2 | 15<=age<25 | 10 |
| R3 | 25<=age<35 | 15 |
| R4 | 35<=age<45 | 20 |
| R5 | 45<=age<65 | 45 |
| R6 | 65<=age<90 | 25 |
| R7 | Age>=90 | 0 |

|  |  |
| --- | --- |
| *Equivalence Class* | *clientAccount.age* |
| EC1 | Age<15 |
| EC2 | 15<=age<25 |
| EC3 | 25<=age<35 |
| EC4 | 35<=age<45 |
| EC5 | 45<=age<65 |
| EC6 | 65<=age<90 |
| EC7 | Age>=90 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Equivalence Class* | *Test Case #* | *Value for testing* | *Expected Output* |
| EC1 | ECT1 | clientAccount.age = -5 | factor = 0 |
| EC2 | ECT2 | clientAccount.age = 20 | factor = 10 |
| EC3 | ECT3 | clientAccount.age = 30 | factor = 15 |
| EC4 | ECT4 | clientAccount.age = 40 | factor = 20 |
| EC5 | ECT5 | clientAccount.age = 55 | factor = 45 |
| EC6 | ECT6 | clientAccount.age = 80 | factor = 25 |
| EC7 | ECT7 | clientAccount.age = 100 | Factor = 0 |

|  |  |
| --- | --- |
| *Equivalence Class* | *clientAccount.balance* |
| EC1 | Balance <= 0 |
| EC2 | 0 < balance < 100 |
| EC3 | 100 <= balance < 500 |
| EC4 | 500 <= balance < 1000 |
| EC5 | 1000<=balance< 2000 |
| EC6 | 2000<=balance<3000 |
| EC7 | Balance >= 3000 |

**Equivalence Class Testing: getBalanceFactor(clientAccount)**

|  |  |  |
| --- | --- | --- |
| *Requirement* | *clientAccount.balance* | *balanceFactor* |
| R1 | Balance <= 0 | 0 |
| R2 | 0 < balance < 100 | 5 |
| R3 | 100 <= balance < 500 | 10 |
| R4 | 500 <= balance < 1000 | 20 |
| R5 | 1000<=balance< 2000 | 60 |
| R6 | 2000<=balance<3000 | 100 |
| R7 | Balance >= 3000 | 0 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Equivalence Class* | *Test Case #* | *Value for testing* | *Expected Output* |
| EC1 | ECT8 | clientAccount.balance = -100 | factor = 0 |
| EC2 | ECT9 | clientAccount.balance = 50 | factor = 5 |
| EC3 | ECT10 | clientAccount.balance = 300 | factor = 10 |
| EC4 | ECT11 | clientAccount.balance = 800 | factor = 20 |
| EC5 | ECT12 | clientAccount.balance = 1500 | factor = 60 |
| EC6 | ECT13 | clientAccount.balance = 2500 | factor = 100 |
| EC7 | ECT14 | clientAccount.balance = 5000 | factor = 0 |

|  |  |
| --- | --- |
| *Equivalence Class* | *accountFactor* |
| EC1 | Factor3= 0 |
| EC2 | 0 < factor3 < 100 |
| EC3 | 100 <= factor3 < 500 |
| EC4 | 500 <= factor3 < 1000 |
| EC5 | 1000 <= factor3 |

**Equivalence Class Testing: accountStatus(clientAccount)**

|  |  |  |
| --- | --- | --- |
| *Requirement* | *accountFactor* | *accountStatus* |
| R1 | Factor3= 0 | “invalid” |
| R2 | 0 < factor3 < 100 | “poor” |
| R3 | 100 <= factor3 < 500 | “acceptable” |
| R4 | 500 <= factor3 < 1000 | “good” |
| R5 | 1000 <= factor3 | “excellent” |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Equivalence Class* | *Test Case #* | *Value for testing*  *getAgeFactor(clientAccount)* | *Value for testing*  *getBalanceFactor(clientAccount)* | *Expected Output* |
| EC1 | ECT15 | getAgeFactor(clientAccount) = 0 | getBalanceFactor(clientAccount) = 10 | accountStatus = “invalid” |
| EC2 | ECT16 | getAgeFactor(clientAccount) = 10 | getBalanceFactor(clientAccount) = 5 | accountStatus = “poor” |
| EC3 | ECT17 | getAgeFactor(clientAccount) = 25 | getBalanceFactor(clientAccount) =10 | accountStatus = “acceptable” |
| EC4 | ECT18 | getAgeFactor(clientAccount) = 10 | getBalanceFactor(clientAccount) = 60 | accountStatus = “good” |
| EC5 | ECT19 | getAgeFactor(clientAccount) = 15 | getBalanceFactor(clientAccount) = 100 | accountStatus = “excellent” |

**Equivalence Class Testing: creditStatus(clientAccount, creditCheckMode)**

|  |  |  |  |
| --- | --- | --- | --- |
| *Requirement* | *creditCheckMode* | *clientAccount.creditScore* | *Output* |
| R1 | “default/strict” | creditScore < 0 | creditStatus = “invalid” |
| R2 | “default/strict” | creditScore < scoreThreshold | creditStatus = “poor” |
| R3 | “default/strict” | creditScore >= scoreThreshold | creditStatus = “good” |
| R4 | “default/strict” | creditScore > 100 | creditStatus = “invalid” |

|  |  |  |
| --- | --- | --- |
| *Equivalence Class* | *creditCheckMode* | *clientAccount.creditScore* |
| EC1 | “strict” | creditScore < 0 |
| EC2 | “strict” | creditScore < 55 |
| EC3 | “strict” | creditScore >= 55 |
| EC4 | “strict” | creditScore > 100 |
| EC5 | “default” | creditScore < 0 |
| EC6 | “default” | creditScore < 85 |
| EC7 | “default” | creditScore >= 85 |
| EC8 | “default” | creditScore > 100 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Equivalence Class* | *Test Case #* | *Value for testing*  *( creditCheckMode)* | *Value for testing*  *( clientAccount.creditScore)* | *Expected Output* |
| EC1 | ECT20 | “strict” | creditScore = -10 | creditStatus = “invalid” |
| EC2 | ECT21 | “strict” | creditScore = 30 | creditStatus = “poor” |
| EC3 | ECT22 | “strict” | creditScore = 70 | creditStatus = “good” |
| EC4 | ECT23 | “strict” | creditScore = 120 | creditStatus = “invalid” |
| EC5 | ECT24 | “default” | creditScore = -5 | creditStatus = “invalid” |
| EC6 | ECT25 | “default” | creditScore = 70 | creditStatus = “poor” |
| EC7 | ECT26 | “default” | creditScore = 95 | creditStatus = “good” |
| EC8 | ECT27 | “default” | creditScore = 120 | creditStatus = “invalid” |

|  |  |  |  |
| --- | --- | --- | --- |
| *Requirement* | *product* | *product Quantity* | *Output* |
| R1 | Invalid name |  | productStatus = “invalid” |
| R2 | Valid name | product Quantity = 0 | productStatus = “soldout” |
| R3 | Valid name | product Quantity < inventoryThreshold | productStatus = “limited” |
| R4 | Valid name | product Quantity >= inventoryThreshold | productStatus = “available” |

**Equivalence Class Testing: productStatus(product, inventory, inventoryThreshold)**

|  |  |  |  |
| --- | --- | --- | --- |
| *Equivalence Class* | *Product =* | *InventoryThreshold* | *productQuantity* |
| EC1 | “non-product” | inventoryThreshold = 200 | product Quantity = 0 |
| EC2 | “productX” | inventoryThreshold = 200 | product Quantity = 0 |
| EC3 | “productY” | inventoryThreshold = 200 | product Quantity = 100 |
| EC4 | “productZ” | inventoryThreshold = 200 | product Quantity = 300 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *Equivalence Class* | *Test Case #* | *Value for testing*  *( product = )* | *Value for testing*  *( inventoryThreshold)* | *productQuantity* | *Expected Output* |
| EC1 | ECT28 | “non-product” | - | - | productStatus = “invalid” |
| EC2 | ECT29 | “productX” | inventoryThreshold = 200 | product Quantity = 0 | productStatus = “soldout” |
| EC3 | ECT30 | “productY” | inventoryThreshold = 200 | product Quantity = 100 | productStatus = “limited” |
| EC4 | ECT31 | “productZ” | inventoryThreshold = 200 | product Quantity = 300 | productStatus = “available” |

**Equivalence Class Testing: orderHandling(clientAccount,product,inventory,inventoryThreshold,creditCheckMode)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Requirement* | *accountStatus* | *creditStatus* | *productStatus* | *Output* |
| R1 | excellent | - | - | orderStatus = accepted |
| R2 | good | good | - | orderStatus = accepted |
| R3 | good | Poor | available | orderStatus = accepted |
| R4 | acceptable | good | available | orderStatus = accepted |
| R5 | Acceptable | Good | Limited | orderStatus = pending |
| R6 | Acceptable | Poor | Available | orderStatus = pending |
| R7 | Poor | Good | Limited | orderStatus = pending |
| R8 | Invalid | - | - | orderStatus = rejected |
| R9 | - | Invalid | - | orderStatus = rejected |
| R10 | - | - | Invalid | orderStatus = rejected |
| R11 | acceptable | Poor | Sold-out | orderStatus = rejected |
| R12 | poor | Good | Sold-out | orderStatus = rejected |
| R13 | Poor | Poor | - | orderStatus = rejected |

|  |  |  |  |
| --- | --- | --- | --- |
| *Equivalence Class* | *accountStatus* | *creditStatus* | *productStatus* |
| EC1 | Excellent | poor | poor |
| EC2 | Good | good | poor |
| EC3 | good | Poor | available |
| EC4 | acceptable | good | available |
| EC5 | Acceptable | Good | Limited |
| EC6 | Acceptable | Poor | Available |
| EC7 | Poor | Good | Limited |
| EC8 | Invalid | Good | good |
| EC9 | Good | Invalid | good |
| EC10 | Good | Good | Invalid |
| EC11 | acceptable | Poor | Sold-out |
| EC12 | poor | Good | Sold-out |
| EC13 | Poor | Poor | available |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *Equivalence Class* | *Test Case #* | *Value for testing*  *( accountStatus() )* | *Value for testing*  *( creditStatus)* | *productStatus* | *Expected Output* |
| EC1 | ECT32 | Excellent | poor | limited | orderStatus = accepted |
| EC2 | ECT33 | Good | good | limited | orderStatus = accepted |
| EC3 | ECT34 | good | Poor | available | orderStatus = accepted |
| EC4 | ECT35 | acceptable | good | available | orderStatus = accepted |
| EC5 | ECT36 | Acceptable | Good | Limited | orderStatus = pending |
| EC6 | ECT37 | Acceptable | Poor | Available | orderStatus = pending |
| EC7 | ECT38 | Poor | Good | Limited | orderStatus = pending |
| EC8 | ECT39 | Invalid | Good | good | orderStatus = rejected |
| EC9 | ECT40 | Good | Invalid | good | orderStatus = rejected |
| EC10 | ECT41 | Good | Good | Invalid | orderStatus = rejected |
| EC11 | ECT42 | acceptable | Poor | Sold-out | orderStatus = rejected |
| EC12 | ECT43 | poor | Good | Sold-out | orderStatus = rejected |
| EC13 | ECT44 | Poor | Poor | available | orderStatus = rejected |

**Boundary Value Testing: getAgeFactor(clientAccount)**

|  |  |
| --- | --- |
| *Equivalence Class* | *clientAccount.age* |
| EC1 | Age<15 |
| EC2 | 15<=age<25 |
| EC3 | 25<=age<35 |
| EC4 | 35<=age<45 |
| EC5 | 45<=age<65 |
| EC6 | 65<=age<90 |
| EC7 | Age>=90 |

|  |  |
| --- | --- |
| *Boundaries* | *clientAccount.age* |
| B1 | 14,15,16 |
| B2 | 14,15,16,24,25,26 |
| B3 | 24,25,26,34,35,36 |
| B4 | 34,35,36,44,45,46 |
| B5 | 44,45,46,64,65,66 |
| B6 | 64,65,66,89,90,91 |
| B7 | 89,90,91 |

|  |  |  |
| --- | --- | --- |
| *Boundary Value Test Case* | *clientAccount.age* | *Expected output* |
| BVT1 | 14 | Factor = 0 |
| BVT 2,3, 4 | 15,16,24 | Factor = 10 |
| BVT 5,6,7 | 25,26,34 | Factor = 15 |
| BVT 8,9,10 | 35,36,44 | Factor = 20 |
| BVT 11,12,13 | 45,46,64 | Factor = 45 |
| BVT 14,15,16 | 65,66,89 | Factor = 25 |
| BVT 17,18 | 90,91 | Factor = 0 |

|  |  |
| --- | --- |
| *Equivalence Class* | *clientAccount.balance* |
| EC1 | Balance <= 0 |
| EC2 | 0 < balance < 100 |
| EC3 | 100 <= balance < 500 |
| EC4 | 500 <= balance < 1000 |
| EC5 | 1000<=balance< 2000 |
| EC6 | 2000<=balance<3000 |
| EC7 | Balance >= 3000 |

|  |  |
| --- | --- |
| *boundaries* | *clientAccount.balance* |
| B1 | -1,0,1 |
| B2 | -1,0,1,99,100,101 |
| B3 | 99,100,101,499,500,501 |
| B4 | 499,500,501,999,1000,1001 |
| B5 | 999,1000,1001,1999,2000,2001 |
| B6 | 1999,2000,2001,2999,3000,3001 |
| B7 | 2999,3000,3001 |

**Boundary Value Testing: getBalanceFactor(clientAccount)**

|  |  |  |
| --- | --- | --- |
| *Boundary Value Test Case* | *clientAccount.balance* | *Expected output* |
| BVT 19,20 | -1,0 | balanceFactor = 0 |
| BVT 21,22 | 1,99 | balanceFactor = 5 |
| BVT 23,24,25 | 100,101,499 | balanceFactor = 10 |
| BVT 26,27,28 | 500,501,999 | balanceFactor = 20 |
| BVT 29,30,31 | 1000,1001,1999 | balanceFactor = 60 |
| BVT 32,33,34 | 2000,2001,2999 | balanceFactor = 100 |
| BVT 35,36 | 3000,3001 | balanceFactor = 0 |

**Boundary Value Testing: Equivalence Class Testing: accountStatus(clientAccount)**

|  |  |
| --- | --- |
| *Equivalence Class* | *accountFactor* |
| EC1 | Factor3= 0 |
| EC2 | 0 < factor3 < 100 |
| EC3 | 100 <= factor3 < 500 |
| EC4 | 500 <= factor3 < 1000 |
| EC5 | 1000 <= factor3 |

|  |  |
| --- | --- |
| *Boundaries* | *Factor3 =* |
| B1 | 0 |
| B2 | -1,0,1, 99,100,101 |
| B3 | 99,100,101,499,500,501 |
| B4 | 499,500,501,999,1000,1001 |
| B5 | 999,1000,1001 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Boundary Value test Case* | *Factor3 =* | *Value for testing*  *getAgeFactor(clientAccount)* | *Value for testing*  *getBalanceFactor(clientAccount)* | *Expected output* |
| BVT37\* | 0 | 0 | 10 | accountStatus = “invalid” |
| BVT38,39 | 1,99 | 1,1 | 1,99 | accountStatus = “poor” |
| BVT40,41,42 | 100,101,499 | 10,101,499 | 10,1,1 | accountStatus = “acceptable” |
| BVT43,44,45 | 500,501,999 | 50, 501, 1 | 10,1, 999 | accountStatus =”good” |
| BVT46,47 | 1000, 1001 | 10, 1 | 100, 1001 | accountStatus = “excellent” |

|  |  |  |
| --- | --- | --- |
| *Equivalence Class* | *creditCheckMode* | *clientAccount.creditScore* |
| EC1 | “strict” | creditScore < 0 |
| EC2 | “strict” | creditScore < 55 |
| EC3 | “strict” | creditScore >= 55 |
| EC4 | “strict” | creditScore > 100 |
| EC5 | “default” | creditScore < 0 |
| EC6 | “default” | creditScore < 85 |
| EC7 | “default” | creditScore >= 85 |
| EC8 | “default” | creditScore > 100 |

**Boundary Value Testing: creditStatus(clientAccount, creditCheckMode)**

|  |  |  |
| --- | --- | --- |
| *Boundaries* | *creditCheckMode* | *clientAccount.creditScore* |
| B1 | “strict” | -1,0,1 |
| B2 | “strict” | -1,0,1,54,55,56 |
| B3 | “strict” | 54,55,56 |
| B4 | “strict” | 99,100,101 |
| B5 | “default” | -1,0,1 |
| B6 | “default” | -1,0,1,84,85,86 |
| B7 | “default” | 84,85,86 |
| B8 | “default” | 99,100,101 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Boundary Value Test Case* | *creditCheckMode* | *clientAccount.creditScore* | *Output* |
| BVT48 | “strict” | -1, | creditStatus = “invalid” |
| BVT49,50,51 | “strict” | 0,1,54 | creditStatus = “poor” |
| BVT52,53,54 | “strict” | 55,56,99 | creditStatus = “good” |
| BVT55,56 | “strict” | 100,101 | creditStatus = “invalid” |
| BVT57,58 | “default” | -1,0 | creditStatus = “invalid” |
| BVT59,60 | “default” | 1,84 | creditStatus = “poor” |
| BVT61,62,63 | “default” | 85,86,99 | creditStatus = “good” |
| BVT64,65 | “default” | 100,101 | creditStatus = “invalid” |

**Equivalence Class Testing: productStatus(product, inventory, inventoryThreshold)**

|  |  |  |  |
| --- | --- | --- | --- |
| *Equivalence Class* | *Product =* | *InventoryThreshold* | *productQuantity* |
| EC1 | “non-product” | inventoryThreshold = 200 | product Quantity = 0 |
| EC2 | “productX” | inventoryThreshold = 200 | product Quantity = 0 |
| EC3 | “productY” | inventoryThreshold = 200 | product Quantity < inventoryThreshold |
| EC4 | “productZ” | inventoryThreshold = 200 | product Quantity >= inventoryThreshold |

|  |  |  |  |
| --- | --- | --- | --- |
| *Boundaries* | *Product =* | *InventoryThreshold* | *productQuantity* |
| EC1 | “non-product” | inventoryThreshold = 200 | 0 |
| EC2 | “productX” | inventoryThreshold = 200 | -1,0,1,199,200,201 |
| EC3 | “productY” | inventoryThreshold = 200 | -1,0,1,199,200,201 |
| EC4 | “productZ” | inventoryThreshold = 200 | -1,0,1,199,200,201 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Boundary Value Test Case* | *Product =* | *inventoryThreshold* | *product Quantity =* | *Expected Output* |
| BVT66 | “non-product” | inventoryThreshold = 200 | 0 | productStatus = “invalid” |
| BVT67,68 | “productA”,“productX” | inventoryThreshold = 200 | -1,0 | productStatus = “soldout” |
| BVT69,70 | “productB”, productC | inventoryThreshold = 200 | 1, 199 | productStatus = “limited” |
| BVT71,72 | “productD”, productE | inventoryThreshold = 200 | 200,201 | productStatus = “available” |

**Decision Value Testing: getAgeFactor(clientAccount)**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **Rules** | | | | | | |
|  |  | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
| **Conditions** | age | age < 0 | 15 < age < 25 | 25 <= age < 35 | 35 <= age < 45 | 45 <= age  < 65 | 65 <= age < 90 | age > 90 |
| **Actions** | Return 0 | x |  |  |  |  |  | x |
| Return 10 |  | x |  |  |  |  |  |
| Return 15 |  |  | x |  |  |  |  |
| Return 20 |  |  |  | x |  |  |  |
| Return 45 |  |  |  |  | x |  |  |
| Return 25 |  |  |  |  |  | x |  |

**Equivalence Class Testing: getBalanceFactor(clientAccount)**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **Rules** | | | | | | |
|  |  | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
| **Conditions** | **balance** | b < 0 | 0 < b < 100 | 100 <= b < 500 | 500 <= b < 1000 | 1000 <= b < 2000 | 2000 <= b < 3000 | b > 3000 |
| **Actions** | **Return 0** | x |  |  |  |  |  | x |
| **Return 5** |  | x |  |  |  |  |  |
| **Return 10** |  |  | x |  |  |  |  |
| **Return 20** |  |  |  | x |  |  |  |
| **Return 60** |  |  |  |  | x |  |  |
| **Return 100** |  |  |  |  |  | x |  |

**Decision Class Testing: accountStatus(clientAccount)**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **Rules** | | | | |  |  |
|  |  | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
| **Conditions** | **Age factor** | age < 0 | 15 < age < 25 | 25 <= age < 35 | 35 <= age < 45 | 45 <= age  < 65 | 65 <= age < 90 | age > 90 |
| **Balance factor** | b < 0 | 0 < b < 100 | 100 <= b < 500 | 500 <= b < 1000 | 1000 <= b < 2000 | 2000 <= b < 3000 | b > 3000 |
| **Actions** | **Return invalid** | x | x |  |  |  |  |  |
| **Return**  **poor** |  | x |  |  | x |  |  |
| **Return acceptable** |  | x |  |  |  |  |  |
| **Return**  **excellent** |  | x | x | x | x | x | x |